Object-Oriented Programming I (CSE 121)
Week # 10
(Inheritance in C++)

Objective(s)

In this lecture, you will learn:

1. What is Inheritance?
2. Why and when to use Inheritance?
3. Program examples of Inheritance
4. Types of Inheritance

Inheritance in C++

Object-Oriented Programming (OOP) has four core concepts which are listed below:

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

In this lecture, we will explore all about inheritance in C++.

Inheritance

- **Inheritance** is one of the key features of OOP in C++. It allows us to create a new class (derived class/child class/sub class) from an existing class (base class/parent class/super class).
- The derived class inherits the features from the base class and can have additional features of its own.

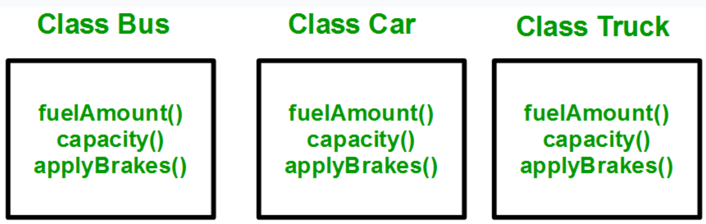
Syntax of Inheritance

```
class parent_class{
    //Body of parent class
};
class child_class : access_modifier parent_class{
   //Body of child class
};
```

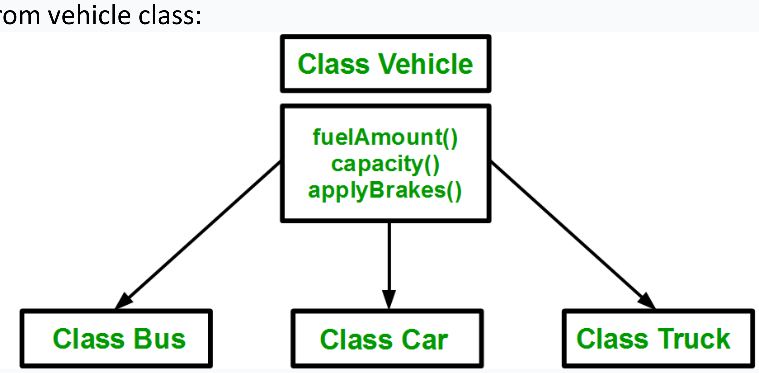
Note: To inherit from a class, use the `:` symbol.

Why and when to use inheritance?

Consider a group of vehicles. You need to create classes for Bus, Car and Truck. The methods fuelAmount(), capacity(), applyBrakes() will be same for all of the three classes. If we create these classes avoiding inheritance then we have to write all of these functions in each of the three classes as shown in below figure:



You can clearly see that above process results in duplication of same code 3 times. This increases the chances of error and data redundancy. To avoid this type of situation, inheritance is used. If we create a class Vehicle and write these three functions in it and inherit the rest of the classes from the vehicle class, then we can simply avoid the duplication of data and increase re-usability. Look at the below diagram in which the three classes are inherited from vehicle class:

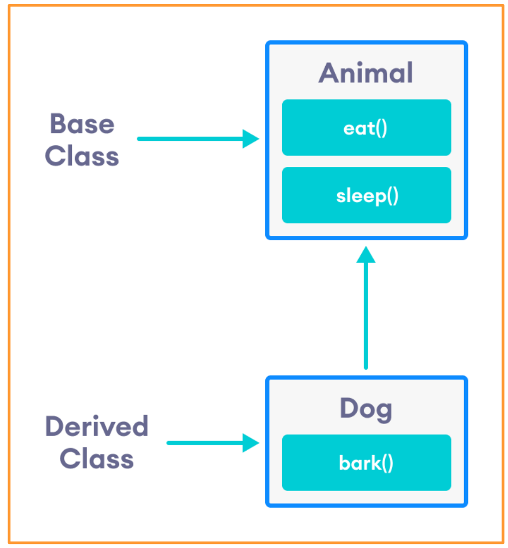


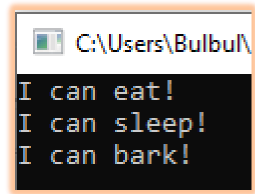
Using inheritance, we have to write the functions only one time instead of three times as we have inherited rest of the three classes from base class(Vehicle).

Program Examples of Inheritance

Example 1: A simple example of Inheritance.

```
#include <iostream>
using namespace std;
// base class
class Animal {
   public:
    void eat() {
        cout << "I can eat!" << endl;
    }
    void sleep() {
        cout << "I can sleep!" << endl;
    }
};
// derived class
class Dog : public Animal {
   public:
    void bark() {
        cout << "I can bark!" << endl;
    }
};
int main() {
    Dog dog1; // Create object of the Dog class
    dog1.eat(); // Calling members of the base class
    dog1.sleep(); // Calling members of the base class
    dog1.bark(); // Calling member of the derived class
    return 0;
}
```





Here, dog1 (the object of derived class Dog) can access members of the base class Animal. It's because Dog is inherited from Animal.

Base Class: The class whose properties are inherited by sub class is called Base Class or Super class or Parent class.
Derived Class: The class that inherits properties from another class is called Derived Class or Sub class or Child class.

is-a relationship

Inheritance is an is-a relationship. We use inheritance only if an is-a relationship is present between the two classes.
Here are some examples:

- A car is a vehicle.
- Orange is a fruit.
- A surgeon is a doctor.
- A dog is an animal.

Example 2: A simple project of Inheritance.

```cpp
#include<iostream>
using namespace std;
//Base class
class Employee{
protected:
    int eId;
    string eName;
    double basic, hr, ma, tax, salary;
    double calculateSalary(int eId, string eName){
        cout << "Salary calculation for employee " << eId << "\n";
        cout << "--------------------------------\n";
        cout << "Employee Name: " << eName << endl;
        cout << "Enter basic salary: ";
        cin >> basic; //10000
        hr = basic * 0.50;//5000
        ma = basic * 0.20;//2000
        tax = basic * 0.10;//1000
        salary = basic + hr + ma - tax;
        return salary;
    }
public:
    Employee(){}
    Employee(int x, string y){
        this -> eId = x;
        this -> eName = y;
    }
    void display(){
        salary = calculateSalary(eId, eName);
        cout << "Salary = " << salary << endl;
    }
};
//sub class
class Programmer : public Employee{
public:
    double bonus = 10000;
    Programmer(int x, string y){
        this -> eId = x;
        this -> eName = y;
    }
    void display(){
        salary = calculateSalary(eId, eName) + bonus;
        cout << "Salary = " << salary << endl;
    }
};
```

```cpp
//sub class
class SystemAnalyst : public Employee{
public:
    double bonus = 30000;
    SystemAnalyst(int x, string y){
        this -> eId = x;
        this -> eName = y;
    }
    void display(){
        salary = calculateSalary(eId, eName) + bonus;
        cout << "Salary = " << salary << endl;
    }
};
int main(){
    Employee e1(1,"Bulbul");
    e1.display();
    cout << "\n\n";
    Employee e2(2,"Aleeza");
    e2.display();
    cout << "\n\n";
    Programmer e3(3,"Esha");
    e3.display();
    cout << "\n\n";
    SystemAnalyst e4(4,"Sharif");
    e4.display();
}
```

C:\Users\Bulbul\Desktop\OOP\Project.exe

```
Salary calculation for employee 1
--------------------------------
Employee Name: Bulbul
Enter basic salary: 30000
Salary = 48000


Salary calculation for employee 2
--------------------------------
Employee Name: Aleeza
Enter basic salary: 20000
Salary = 32000


Salary calculation for employee 3
--------------------------------
Employee Name: Esha
Enter basic salary: 25000
Salary = 50000


Salary calculation for employee 4
--------------------------------
Employee Name: Sharif
Enter basic salary: 40000
Salary = 94000
```

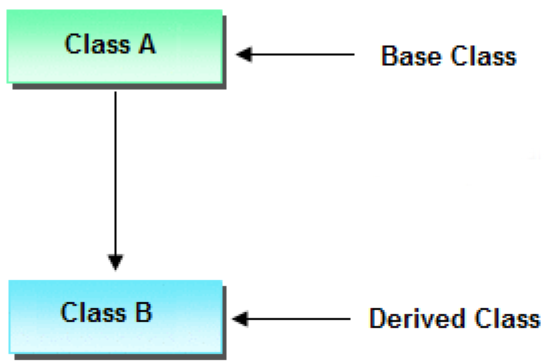
Types of Inheritance

In C++, we have 5 different types of Inheritance. Namely,

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance (also known as Virtual Inheritance)

Single inheritance

In this type of inheritance one derived class inherits from only one base class. It is the most simplest form of Inheritance.

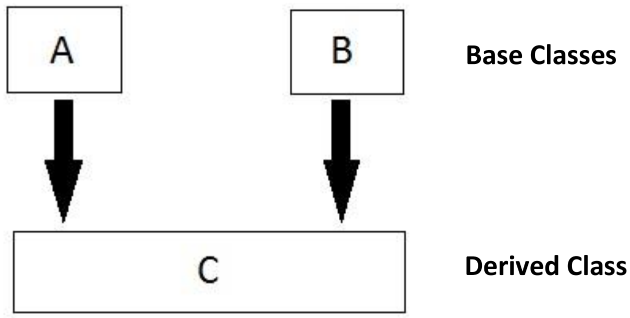


Example:

```
#include <iostream>
using namespace std;
//Base or parent or super class
class Shape{
  public:
    void setWidth(int w){
      width = w;
    }
    void setHeight(int h){
      height = h;
    }
  protected:
    int width;
    int height;
};
//derived or child or sub class
class Rectangle : public Shape{
  public:
    int getArea(){
      return width*height;
    }
};
int main(){
  Rectangle rec;
  rec.setWidth(10);
  rec.setHeight(20);
  cout << rec.getArea();
  return 0;
}
```

Multiple inheritance

In this type of inheritance, a single derived class may inherit from two or more than two base classes.



Example:

```
#include <iostream>
using namespace std;

//base or parent or super class
class Shape{
   public:
      void setWidth(int w){
         width = w;
      }
      void setHeight(int h){
         height = h;
      }
   protected:
      int width;
      int height;
};

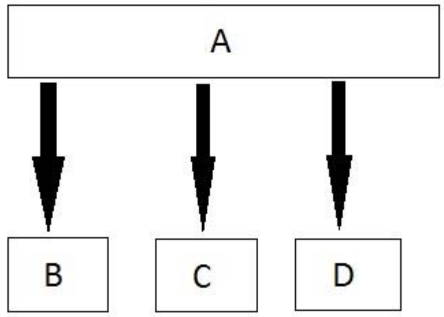
class paintCost{
   public:
      int getCost(int area){
         return area * 70;
      }
};

//derived or child or sub class
class Rectangle:public Shape, public paintCost{
   public:
      int getArea(){
         return width * height;
      }
};
```

```
int main(){
   Rectangle rec;
   int area;
   rec.setWidth(5);
   rec.setHeight(10);
   area = rec.getArea();
   cout << "Total Area : " << area << endl;
   cout << "Total Cost : Tk." << rec.getCost(area);
   cout << endl;
   return 0;
}
```

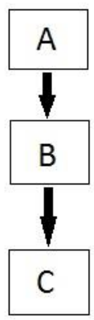
Hierarchical Inheritance

In this type of inheritance, multiple derived classes inherit from a single base class.



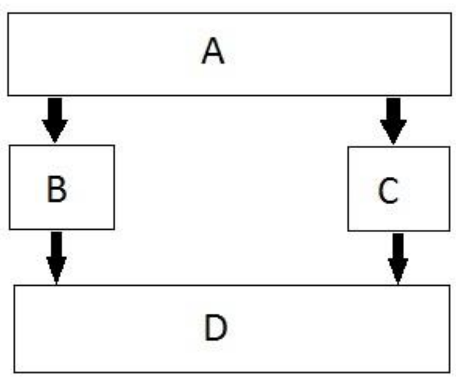
Multilevel Inheritance

In this type of inheritance, the derived class inherits from a class, which in turn inherits from some other class.



Hybrid (Virtual) Inheritance

Hybrid Inheritance is combination of Hierarchical and Multilevel Inheritance.



....